DESIGN AND MANAGEMENT OF A SECURE NETWORKED ADMINISTRATION SYSTEM : A PRACTICAL APPROACH

Vijay Varadharajan
Professor of Computing, University of W. Sydney, Australia.
email : vijay@st.nepean.uws.edu.au

June 7, 1996

Abstract

As applications become more distributed, the design and management of security services in networked systems play an increasingly significant role. This paper describes the design of services for securing the management of a networked administration system. It presents the architectural principles involved and the overall security solution comprising the design of security services and the trusted components that provide these services. The security schemes are illustrated by providing a walkthrough of the networked administration scenario.

1. Introduction

Security plays a vital role in the design, development and practical use of the distributed computing environment, for greater availability and access to information in turn imply that distributed systems are more prone to attacks. The need for practical solutions for secure networked system management is becoming increasingly significant. In developing these solutions, several important issues need to be carefully addressed. The design of the required security services forms a major part. Often the issues associated with security management are not adequately addressed. First, it is important to identify clearly the functionalities and interfaces of the trusted security management components. Then it is necessary to consider whether some of these trusted management authorities can be grouped together to simplify the overall management. This depends on several factors such as the relationships between organizations (or units) involved in the networked environment and the types of services offered as well as performance considerations. In practice, it is also necessary to consider the system development and deployment in stages thereby enabling a staged adoption.

In this paper, we address the design and management of a secure networked administration system. The paper is organized as follows: Section 2 describes a network administration scenario, and outlines the different stages involved in the development of the system. Section 3 discusses the architectural issues and outlines the design of security services and the provision of security facilities. The secure system operation is described in Section 4. We outline the different phases involved in the life of a user, application and the system, and describe how the security services are managed by the various components in the architecture. Finally Section 5 provides a walkthrough of the network administration scenario and illustrates the use of security services and facilities.

2. Secure Networked Administration System Design

2.1 Scenario

The scenario we consider is an example demonstration of a secure distributed application. The scenario involves administration of multiple hosts in a network using a single administration station from which

cation that we consider is a distributed configuration and auditing of networked systems. It involves such tasks as configuration of audit scripts (for instance, specifying what checks to be done), collection of audit information, and browsing the audit data.

In a large practical networked environment, there will be several managers responsible for different parts of the network. Our scenario allows different security managers to have different sets of privileges. For instance, Security Manager A might be responsible for hosts 1,2, 3 and 4, and might have authority to configure, audit and browse audit information of hosts 1,2 and 3, and only browse audit information of host 4, whereas Security Manager B is responsible for hosts 3 and 4, and has authority to configure, audit and browse host 4 and only has browse authority for host 3. More generally the privileges capture both geographical partitioning of the networked hosts as well as the type of actions that a manager can perform over the hosts.

To ensure that only authorized entities are able to set the configurations and control the audit process, it is necessary to provide mutual authentication between the security administration agents and the remote hosts. Furthermore, secure transfer of information between remote hosts and the security administrator workstations is required. Hence this scenario brings together issues of privilege control, authentication, secure communication and auditing in an integrated manner.

In addition, the task of administering networked systems is a round the clock activity. Hence it may be necessary for the security manager to access the security administration workstation remotely, e.g. from home or from a different location in the site. For instance, the manager may browse through the security status of the network system before determining whether a visit to the site is required. However the set of privileges that a manager has while accessing from a remote location is likely to be different from those that she has while physically present at the administration workstation[1]. Our scenario envisages secure remote access using a mobile personal information appliance such a palmtop computer over either a public switched telephone network or a wireless network.

The major stages of the Secure Networked Administration System (SNAS) development are (See Figure 2) :

(a) from a single Security Administration Station (SAS) with a single Security Manager.

(b) from a single SAS with mutiple Security Managers responsible for different parts of the network system, and having different sets of privileges.

(c) from a single SAS with mutiple Security Managers, with remote access to SAS from a mobile device (dial in/wireless).

(d) with mutiple SAS - one SAS per domain. (A domain comprises a collection of hosts over which a single SAS has jurisdiction).

3. Secure System Design : Architectural Issues

3.1 Design Goals

The basic set of design goals, related both to the definition of the services provided by the components and their implementation are as follows :

- With respect to the development of such a secure system, the aim is not to produce a monolith. We consider this to be in phases thereby enabling a staged adoption.

[1] For instance, this could be a proper subset.

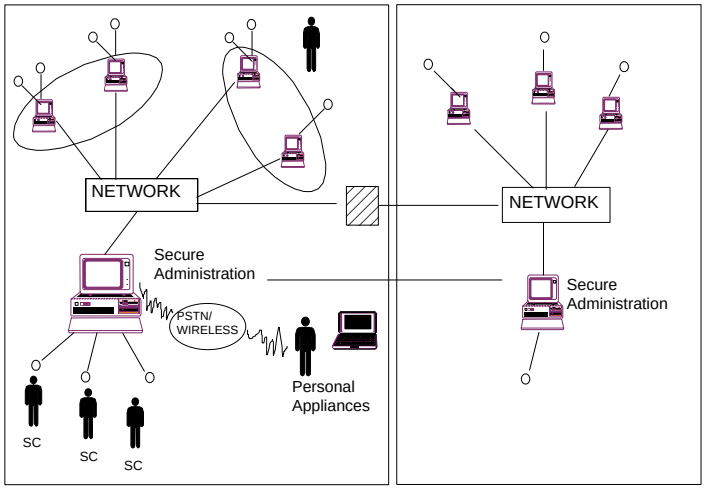

Figure 1: Network Administration Scenario

- Uniform treatment of agents acting as principals, no matter what kind of agents they are (person, hardware or software component)
- The implementation of components will be heavily dependent on the operating system interfaces. However the model of the operating system that we have assumed applies to a broad range of hosts, allowing re-implementation of the same service definitions and protocols as necessary.
- The choice of cryptographic algorithms is an important one due to licensing and export control issues, as well as technological feasibility. This is not a question of providing many protocols, but of implementing them behind uniform service definitions, so that the application developer can work independently of this decision.
- Support management of security information wherever it is distributed, not just at a central location. Also, the aim is to bring the choice of mechanisms behind the service definitions into the management world, not forcing the application developer to hardwire them.
- Integration of security management with network and system management, thereby providing a uniform management view to the administrator.

3.2 SNAS Services and Facilities

The security services provided by SNAS are the following:

- *Authentication Service* : This service supports authentication of both interactive (e.g. a human user) and non-interactive principals (e.g. applications) [7].
- *Authorization Service* : This service allows an application to decide whether a request for a particular service by another principal is to be granted or not [8].
- *Secure Communication Service* : This service provides secure communication of information transferred between remote principals. Secure communication here implies confidentiality, integrity or both.
- *Auditing Service* : The auditing service considered in SNAS provides a snapshot of the system at a given time, thereby allowing a security administrator to easily inspect the security status of the system.

One may view this approach as an extension of the Kerberos [3] and DCE [4] systems, which are at present based on symmetric key technologies. The DCE is planning to introduce the public key technology in an incremental manner. A version of public key based Kerberos has also appeared in [12].

We also introduce the concept of an Authorization Server which captures more sophisticated access control information compared to the Privilege Server in the DCE (which primarily deals with groups). The access control information that we consider have different static and dynamic characteristics. Role is an example of such access control information. More significantly, the architectural as well as the design issues described in this paper should be relevant to future DCE extensions.

We now describe the design and operation of these services by considering

- the trusted components of the architecture that are involved in the provision of these services,
- the security information and attributes used by these services and where they are stored and how they are distributed, and
- the different phases involved in the life of a user, application and the system.

3.3 Principals

Principals are the basic elements over which access control can be exercised. A principal is the smallest entity that can be authenticated across a collection of machines in a domain. Thus, for a domain comprising Unix machines, a principal is a map from machines to UIDs.

Let us now consider the trusted principals that exist in our architecture.

We have a single *Certification Server* (CS) principal, which is a global entity in a domain, and an *Authentication Server Component* (ASC) principal on each machine. The CS retains keys associated with the principals and the ASCs. To avoid the need to securely install the key of every principal in the database of every other principal, the CS has been provided.

We have a *Rolebase Server* (RS) principal. For the moment, we will assume one such entity per domain, though there is no reason why there should not be several such entities. The RS has information on which users (principals) have what roles in the domain. E.g. a user Fred is a accountant in organization X. This role information is assumed to be of a general type. We have an *Authorization Broker* (AB) principal on each machine. AB performs the following functions. First, it provides an application principal in a machine the role of a user who is binding to the application. AB obtains this information by contacting the RS. Secondly the AB at the target end verifies the authenticity of the role information provided by the client. Thirdly, at the target end, AB checks the access control information (ACI) – which privileges what users (based on ids and roles) have –, and advises the target application on whether to grant the request or not. The ACI is stored at the target.

Hence we have the following trusted principals (See Figure 2):

- For Authentication Service
  - Authentication Server Component (ASC)
  - Certification Server (CS)
- For Authorization Service
  - Authorization Broker (AB)
  - Rolebase Server (RS)

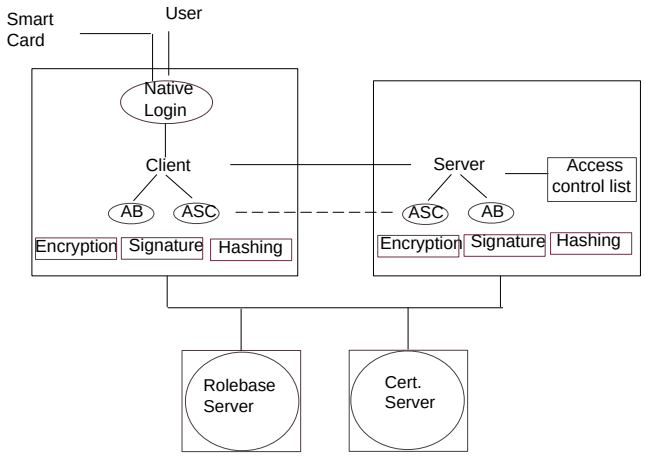

Figure 2: Security Components and Trusted Authorities

3.4 Security Information in SNAS

There are two types of security information involved in these various phases in SNAS, namely that are stored in various security components and that are transferred between components.

Let us consider the characteristics of the different types of security information. Some security information are of *generic* and *static* in nature. Identity based authentication information typically falls into this category. Some security information are *specific* and still somewhat *static* in nature. Role based information falls into this category. Roles are specific to organizations and they are reasonably static in the sense that they are unlikely to change on a day to day or even on a monthly (or even yearly) basis. In fact, one of the main benefits of the role based access control is to reduce the effect of the changes in the user population on the management of access privileges. Then we have security information that are *specific* and *dynamic* in nature. Specific in the sense that they may relate to applications and/or parts of applications such as files. They tend to be dynamic in the sense they are prone to changes as and when updates are made to applications and functionality changes occur.

Furthermore, the authorities involved in the management of these different types of security information are likely to be different. Not only the strategies with respect to *when* the changes and updates to these information take place are likely to be different (mentioned above) but also *who* are allowed to make these changes are likely to vary. For instance, the specification and changes to the role information in an orgnization will be the responsiblity of a certain group of people who can be different to those responsible for setting the privileges for a specific file or application in a server.

3.5 Design Principles

From an architectural point of view, such a characterization leads to the following design principles [8].

Principle 1

Store the static and generic information in some form of a central server responsible for a collection of clients and servers (in a domain).

Principle 2

Store the dynamic and specific information near or in the end system where the target applications reside, enabling the end system authorities to be involved in their management.

The above characterization also affects the way the security information is being distributed.

Static and generic information, being stored in a central server in a domain, can be "pushed" by the client to the target application server. In fact, static and specific information can also be "pushed" in a similar fashion.

Principle 4

Specific and dynamic information needs to be "pulled" at the time of the decision process.

It is important to note that these two types of information may be stored in two different servers owned and managed by different authorities. Based on these principles, one can certainly argue for the need for two trusted authorities —- one dealing with generic and static security information and the other dealing with specific and static security information — both of which can be architected as central servers servicing multiple clients and servers within a domain. These two correspond to the Certification Server and the Rolebase Server in our architecture. The Certification Server stores the authentication certificates of the principals, which are static and generic. The Rolebase Server stores the user identities and the roles (and their generic privileges) that can be taken by these identities. These are organization specific and are still relatively static.

The target server stores specific and dynamic security information; often such information are dependent on the state of the application or resource under consideration. Such information include attributes associated with specific rights in the application. For instance, the client might be allowed to withdraw 10000 dollars from Monday to Friday. He might have withdrawn 4000 dollars on Monday, leaving her only 6000 dollars for the rest of the period. So when the client makes subsequent requests, the previous state associated with the transaction needs to be taken into account.

The system's security information is captured using the following constructs:

- A Certificate containing the identity and the public key related information transferred from the Certification Server to the Authentication Server Component. This is signed using the private key of the CS.
- An Authentication Token between the client and the server ASCs for mutual authentication. This is protected using the public key of the target (or client) and signed using the private key of the client (target).
- A Token containing the identity and role information transferred from the Rolebase Server to the Authorization Broker. This token is signed by the Rolebase Server using its private key.
- Access Control Information representing the dynamic and specific information and state dependent information residing at the target end systems.
- Secure conversation between client and target principals, protected using symmetric conversation key established at the end of the mutual authentication process.

3.6 Authorization Service Design

The design of authorization service for distributed applications is an important topic and it merits a separate paper of its own which is in preparation [8]. Here we outline some of the relevant features that form part of the Rolebase Server and the Authorization Broker in SNAS.

The administrator of a networked system in an orgnaization needs to manage privileges of individuals in terms of group profiles, department membership and so on. Furthermore the "give" rights of various administrators need to be configured. Hence the need for a policy language. The policies expressed in this language must be translated into a form usable by the Authorization Broker at access decision time. In particular, the representation of the policies at administration time at the Rolebase Server is likely to be different from the representation of the policy at runtime used by the Authorization Broker.

of the Rolebase Server:

- An Administration Store : Stores the policy expressions whose interface allows an administration client (user) to input and modify policy statements.
- An Evaluation Store : Stores the policies expressions in a representation suitable for runtime access and decision. As mentioned above, this is different from the administration policy representation in that here one can compile out the semantics of inheritance and overrides in the expressions, thereby making the access decision faster, for instance, by avoiding the need to search the inheritance hierarchy.
- A compiler that translates the policy expressions from the administration time representation to evaluation time representation.
- An engine that evaluates and services a query, and encapsulates the privilges in the form of an Authorization Token and passes it to the requesting client. The Token is passed to the Authorization Broker of the Server which interprets and evaluates the authorization infomation along with its locally stored ACI to make the access decision.

4. System Operation

We present the characteristics of the system by outlining the operations involved in the different system phases.

4.1 Phases

We identify the following phases in the system.

4.1.1 Installation Phase

In the **Installation phase**, we assume that all the required software components of SNAS are correctly installed. We will assume that the Rolebase is also initialized. We will also assume that the access control lists and the mapping from roles to privileges at the (target) servers have also been initialized.

4.1.2 Certification Phase

In the **Certification** phase, the principals are identified to the CS and the keys associated with them are registered with the CS. In the case of machine principals, the keys are public keys, and the CS creates certificates. A certificate comprises the name, the Id, the public key of the principal, and a validity period, signed by the CS's private key. Hence CS stores certificates of ASCs of different machines (including Rolebase Server). We assume that the public keys of the Certification Server and the Rolebase Server are known to all ASCs in the system. For users with smartcards, we can store the private keys in the smart card. If the smart card technology only allows symmetric key based computation then we have the secret symmetric key of the user stored securely in the smartcard and in the CS.

4.1.3 Booting Phase

In the **Booting** phase, when a machine is switched on, the ASC of that machine authenticates itself to the CS using a challenge-response protocol. The CS sends a challenge to which the ASC produces a response using his private key of the public key system. Recall that the ASC has registered its public key with the CS during the certification phase. Following a successful challenge-response protocol, a connection number is established between the ASC in the machine and the CS, which is subsequently used when a principal (user or an application) in that machine requires information from the CS.

Consider the situation where a user U wishes to log on to a machine X, and an application A_x in machine X acting on behalf of user U invokes an application B_y in machine Y for a service. A_x is acting as a client and B_y is acting as a server.

Let us first consider the **Session** phase. In this phase, an agent acting as a principal presents itself to the system : in effect, to the CS. In the case of users, this process involves a login facility and may involve the smartcard, if this is being used.

Following the certification phase, recall that both the public key of the AS in machine X and the secret symmetric key of the user smartcard have been registered with the CS.

The challenge-response protocol to establish the initial user authentication as follows :

The user logs on by providing his Id and his PIN. The login facility passes this information to the smartcard (SC) which checks the validity of the PIN. The use of the login initiates a session with the ASC on the machine. The ASC now sends the principal Id to the CS, signed by the private key of the ASC. The CS generates a fresh nonce as a challenge and the corresponding response using the user secret symmetric key. The challenge-response pair is then signed using the private key of the CS and sent to the ASC. Now the ASC passes the challenge to the SC (via the login facility). The SC calculates the response and sends this to the login which is then able to verify by matching it with the one received from the CS.

When a principal in machine X (e.g. A_x) wishes to request a service from another principal (e.g. B_y) on the remote machine Y, their respective ASCs will need to communicate. If it is the case that the ASC of machine X is not aware of the ASC of machine Y, then it will make use of the CS Certification Server as a directory to obtain the certificate containing the public key of B's ASC. (Once an ASC has obtained a certificate, this can be cached.) Now using the public key of Y's ASC, X's ASC can establish a conversation key which is used in the protection of communications between the principals A and B. This phase is referred to as the **Binding** phase, which concludes with the establishment of a secure channel between the client (A_x) and the server (B_y).

Then comes the **Request** phase where A_x makes the request for a service to B_y using the established secure channel. Before this happens, the client A_x talks to the Authorization Broker (AB) principal to find out the role of the user who is binding to it. It provides AB the authenticated Id of the user. AB then has a conversation with the Rolebase Server machine. Note that this conversation needs only to be protected for integrity and authenticity and not for confidentiality. This is because the user to role mapping is not likely to be sensitive information. The Role Token captures the user Id, the Role information and its associated privileges along with timestamps. This information needs to be verified by the AB of the target server. These requirements are met using a public key based protocol between the AB and the RS. Recall that following the certification phase, the public key of RS is known to AB.

In the **Message phase**, peer to peer communication between the principals A and B occur. These messages are protected using the conversation key established above. Note that we have a different conversation key whenever a new binding between two ASCs occur. For instance, if two principals A_x and B_y complete one conversation and then have a second conversation, then the conversation key would be different in the two cases. Protection here could be just confidentiality (using encryption), or integrity (using cryptographic checksums), or both. The ASC and CS are not involved. The secure communications facility allows the application programmer to set up such connections and use the agreed algorithms and keys transparently, as a secure version of TCP.

B_y now has to decide whether or not to grant the request from A_x. B_y requests the AB to verify the claimed role of the user who is making the request via A_x. AB communicates with the access control information (ACI) database, which contains information on what privileges are allowed for what user identities and roles, and for what applications. At present, we assume that this ACI resides locally on the target machines. AB interprets this information and advises the application on whether to grant

A full description of the security protocols involved in the different phases above can be found in [9].

5. Example Walkthrough

Let us now return to the network administration example.

We have a Security Administration (SAS) station which runs management applications for configuring, collecting, analysing and presenting audit information in a networked system. It provides secure administration of network of Unix systems from a single management station by authorized users - security and network system administrators. From this central station, the security administrator can easily evaluate the level of security at remote systems. It provides quick inspection of security status of the networked system and helps to maintain a minimum level of security. In particular, it is intended to provide snapshots of the system at chosen times to point out existing security anomalies (cf. health checks).

There are audit agent applications residing in each of the remote system that needs to be administered. User U logs onto the secure administration (SAS) workstation, and invokes an audit management application A. The audit management application, acting on behalf of U, requests service from a remote audit agent application B residing in one of the hosts to be administered. We will refer to this host as Y. The request could involve configuration of audit files and audit checks in remote audit agents, activation of audit agents, and transfer of information pertaining to the security status of the remote host and related audit data.

The user U with the smart card is first authenticated using the ASC of the SAS and the Certification Server. The ASCs of the SAS and Y communicate to mutually authenticate each other and establish a common conversation key. This establishes a secure channel between the audit management application A and the remote audit agent B.

The next step is to establish the privileges of the user in question, using the procedures described above. This involves the Authorization Broker of the SAS communicating with the Rolebase Server to determine the role and privileges of the user U. This is used to establish the fact that the user U can have an administrator role and determine the generic privileges associated with this role. The signed role information and the certified identity information (obtained from the Certification Server) are passed to the remote audit agent B, along with the request. The relevant parts of the communication are protected using the previously established conversation key between A and B. B now requests its Authorization Broker (AB) to verify the claimed role of the user making the request and determine the access rights using the Access Control Information (ACI) database. AB interprets this information and advises the audit agent application B on whether to service the request from A.

5.1 Specific Implementation Choices

In this particular application, the SAS performs the administration functions for a networked system of clients and servers. Given this role for the SAS, it is natural for it to hold the role and access privilege information. That is, an implementation choice is made to co-locate the Rolebase Server with the SAS. Note that from the design point of view, the interfaces of the Rolebase Server remain the same. However with this implementation it is not necessary to protect the communication channel between the AB and the RS as they occur within the system.

Stage (b) of SNAS specifies privileges of the various Security Managers in its Rolebase Server. The privilege expressions capture both the range of hosts and subnets that are to be managed by a Security Manager as well as the classes of actions that the Manager has the authority to perform. For instance,

and 4 in Subnet N2, AND *Audit all Hosts in Subnet N2.*

- Manager B can *Configure and Audit all Hosts in Subnet N1.*

The language used to specify the privileges, and their representation and management is described in [8].

Stage(c) involves two additional aspects. The first aspect is the authentication of the remote user and the mobile device over a wireless or dial-in connection. Challenge-response technique similar to the one described earlier has been used to achieve this. Hence we will not describe this here. The second aspect concerns the difference in privileges of a Security Manager when accessing the SAS remotely over a wireless network using a mobile appliance compared to the same Manager accessing the SAS while physically present at the administration workstation. This difference in privileges is captured as part of the policy specifications in the Rolebase Server residing within the SAS. Once again, the language constructs have been designed in such a way to cater for these situations.

Regarding the cryptoalgorithms, appropriate choices are 512-bit RSA for public key based authentication, DES for encryption of data communications, and MD5 for generating hashed message digests.

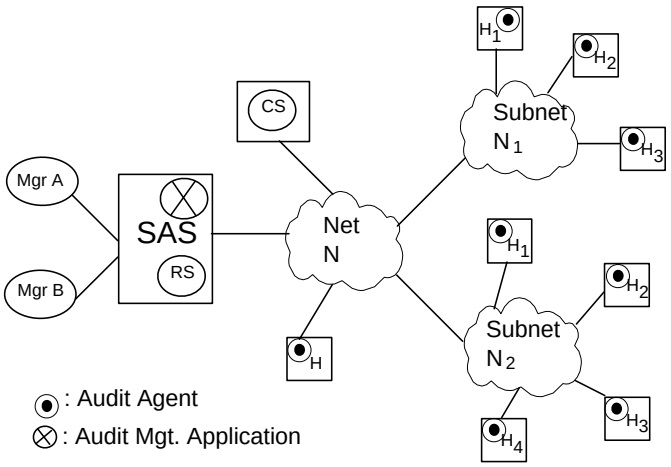

Figure 3: Secure Network Administration System

Acknowledgements : The author would like to thank the anonymous referees for their valuable comments.

References

[1] Y.Yemini, "Emerging Trends in Networks and System Management", Third International Symposium on Integrated Network Management, San Francisco, USA, 1993.

[2] M.Gasser et al, "The Digital Distributed System Security Architecture", National Computer Security Conference, Baltimore, USA, 1989.

[3] Clifford Neumann, Ts Theodore, Kerberos : An Authentication Service for Computer networks, IEEE Communications, Vol.32, No.9, Sept.1994.

[5] J.J.Tardo, K.Alagappan: "SPX: Global Authentication using Public Key Certificates", Proc. of the IEEE Conference on Security and Privacy, 1991.

[6] V.Varadharajan, P.Allen, S.Black, "An Analysis of the Proxy Problem in Distributed Systems", Proc. of the 1991 IEEE Symposium on Research in Security and Privacy, 1991.

[7] Lampson et al, "Authentication in Distributed Systems : Theory and Practice", ACM Trans. on Computing Systems, Vol.10, No.4, 1992.

[8] V.Varadharajan, "Authorization in Distributed Systems", In preparation. Abstract was presented at 1995 IEEE Symposium on Security and Privacy, Oakland, as a Short Presentation.

[9] V.Varadharajan, "Design of a Secure Network Administration System", Technical Report, UWS Computing, 1995.

[10] V.D.Gligor, S.W.Luan, J.N.Pato, "Om Inter-Realm Authentication in Large Distributed Systems", Proceedings of the IEEE Conference on Security asnd Privacy, 1992.

[11] R.Yahalom, B.Klein, T.Beth, "Trust based Navigation in Distributed Systems", Computing Systems, Vol.7, No.1, 1994.

[12] R.Ganesan, "Augmenting Kerberos with Public Key Cryptography", ISOC Symposium on Network and Distributed System Security, 1995.